Numerical Methods of Thermo-Fluid Dynamics II

Summer Semester 2021

Deliverable Task II:

Implementation of Continuous Species Transfer (CST) in interFoam

Submitted by:

Serouj Djabraian
Matrikelnummer 22849126
serouj.d.djabraian@fau.de

Franciskus Xaverius Erick
Matrikelnummer 22811754
franciskus.erick@fau.de

Prashanth Prakash Kamath
Matrikelnummer 22835087
prashanth.kamath@fau.de

John Sebert Mampilli
Matrikelnummer 22833931
john.mampilli@fau.de

Sudesh Rathnayake Mudiyanselage
Matrikelnummer 22849910
sudesh.rathnayake@fau.de

Introduction

The objective of Deliverable Task II was to implement the continuous species transfer (CST) equation in the interFoam solver of OpenFOAM. The CST equation is:

$$\frac{\partial C_i}{\partial t} + \nabla \cdot (C_i U) = \nabla \cdot (D_a \nabla C_i) - \nabla \cdot \left(\gamma_f D_h \frac{1 - {}^{1}\!/\!{}_{H}}{\alpha_1 + {}^{(1-\alpha_1)}\!/\!{}_{H}} C_i \nabla \alpha_1 \right)$$

$$+ \nabla \left[(1-\gamma_f)(D^1 - D^2)\alpha_1 \left(\frac{1}{\alpha_1 + {}^{(1-\alpha_1)}\!/\!{}_{H}} - 1 \right) \nabla C_i \right]$$

$$+ \nabla \left[(1-\gamma_f) \frac{C_i}{\alpha_1 + {}^{(1-\alpha_1)}\!/\!{}_{H}} \left(\frac{1}{H} \frac{D^1 - D^2}{\alpha_1 + {}^{(1-\alpha_1)}\!/\!{}_{H}} - \left(D^1 - \frac{D^2}{H} \right) \right) \nabla \alpha_1 \right]$$

where:

- C_i is the species concentration in mol/m^3,
- $D_a = D^1 \alpha_1 + D^2 \alpha_2$, arithmetic mean mixture diffusivity,
- $D_h = \dfrac{D^1 D^2}{\alpha_1 D^2 + \alpha_2 D^1}$, harmonic mean mixture diffusivity,
- $\gamma_f = |n_f \cdot n_\Sigma|, n_f = \dfrac{S_f}{|S_f|}, n_\Sigma = \dfrac{\nabla \alpha_1}{|\nabla \alpha_1|}$
- $H = \dfrac{C_g}{C_l}$, Henry's Law constant,
- D^1 and D^2 are the diffusion coefficients of the species in the gas and liquid phases respectively.

Implementation of CST Equation in interFoam Solver

The interFoam solver was modified to implement the CST equation. The following sections detail the changes and additions made to the source code. The implementation compiles successfully using OpenFOAM v1906 and v2106; compilation on OpenFOAM v6 and OpenFOAM-dev was found to return errors, likely due to differences between the two (OpenCFD/ESI and OpenFOAM Foundation) distributions.

CEqn.H

The CST equation was implemented in a separate file, *CEqn.H*, as shown below:

```
/*
 *  NMTFD-2 Deliverable Task II: Continuous Species Transfer
 */

const dimensionedScalar deltaN1("deltaN1",1e-8/pow(average(alpha1.mesh().V()), 1.0/3.0));

volScalarField D_harmonic =  (D1*D2)/(alpha1*D2 + (1-alpha1)*D1)  ;

volScalarField D_arithmetic  = alpha1*D1 + (1-alpha1)*D2 ;

dimensionedScalar D1_D2 = D1-D2;



const surfaceVectorField& Sf = mesh.Sf();
const surfaceScalarField& magSf = mesh.magSf();
const surfaceVectorField N_hat_s(Sf/magSf);
const volVectorField gradAlpha1(fvc::grad(alpha1));
const surfaceVectorField gradAlphaf1(fvc::interpolate(gradAlpha1));
const surfaceVectorField n_hat_f(gradAlphaf1/(mag(gradAlphaf1) + deltaN1));
const surfaceScalarField gamma_f = N_hat_s & n_hat_f;
const surfaceScalarField gamma_f_magnitude = mag(gamma_f);

// Coefficients to simplify CST Equation
```

```

volScalarField beta = 1/(alpha1 + ((1-alpha1)/H));
volScalarField K4 = D_harmonic * beta * (1-1/H);
volScalarField K5 = D1_D2 * alpha1 * (beta-1);
volScalarField K6 = beta * ((1/H*beta*D1_D2) - (D1-D2/H));

// Interpolating coefficients for fvm::div() and fvm::laplacian

surfaceScalarField psi4 = gamma_f_magnitude * fvc::interpolate(K4) *
    fvc::snGrad(alpha1) * mesh.magSf();
surfaceScalarField psi5 = (1-gamma_f_magnitude) * fvc::interpolate(K5);
surfaceScalarField psi6 = (1-gamma_f_magnitude) * fvc::interpolate(K6)
    * fvc::snGrad(alpha1) * mesh.magSf();

// CST Equation

fvScalarMatrix CEqn
(
      fvm::ddt(C)
    + fvm::div(phi,C)
    - fvm::laplacian(fvc::interpolate(D_arithmetic), C)
    + fvm::div(psi4, C)
    - fvm::laplacian(psi5, C)
    - fvm::div(psi6, C)
);

Info<< "\nSolving for C" << endl;
CEqn.solve();

Info<< "Min C: " << min(C).value() << endl;
Info<< "Max C: " << max(C).value() << endl;
```

Listing 1: CEqn.H

createFields.H

The following lines were added to createFields.H, to read the coefficients D^1, D^2 and H and create *volScalarField* C:

```
// Declare dictionary phaseProperties for diffusivities and Henry's Law
    Constant
IOdictionary phaseProperties
(
    IOobject
    (
        "phaseProperties",
  runTime.constant(),
  mesh,
  IOobject::MUST_READ,
  IOobject::NO_WRITE
    )
);

// Reading phase diffusivities and Henry's Law Constant
const dimensionedScalar D1("D1", dimViscosity, phaseProperties);
const dimensionedScalar D2("D2", dimViscosity, phaseProperties);
const dimensionedScalar H("H", phaseProperties);

// Declare species concentration field C
(
    IOobject
    (
        "C",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

Listing 2: createFields.H

alphaEqnSubCycle.H

The new file *CEqn.H* was included in the main solver in the existing file *alphaEqnSubCycle.H*:

```
if (nAlphaSubCycles > 1)
{
    dimensionedScalar totalDeltaT = runTime.deltaT();
    surfaceScalarField rhoPhiSum
    (
        IOobject
        (
            "rhoPhiSum",
            runTime.timeName(),
            mesh
        ),
        mesh,
13         dimensionedScalar("0", rhoPhi.dimensions(), 0)
14     );
15 
16     tmp<volScalarField> trSubDeltaT;
17 
18     if (LTS)
19     {
20         trSubDeltaT =
21             fv::localEulerDdt::localRSubDeltaT(mesh, nAlphaSubCycles);
22     }
23 
24     for
25     (
26         subCycle<volScalarField> alphaSubCycle(alpha1, nAlphaSubCycles)
       ;
27         !(++alphaSubCycle).end();
28     )
29     {
30         #include "alphaEqn.H"
31         #include "CEqn.H"
32         rhoPhiSum += (runTime.deltaT()/totalDeltaT)*rhoPhi;
33     }
34 
35     rhoPhi = rhoPhiSum;
36 }
37 else
38 {
39     #include "alphaEqn.H"
40     #include "CEqn.H"
41 }
42 
43 rho == alpha1*rho1 + alpha2*rho2;
```

Listing 3: alphaEqnSubCycle.H

Lines 31 and 40 were added to *alphaEqnSubCycle.H* to include the CST equation to the interFoam solver.

Case Setup

The modified interFoam solver was used to simulate the rise of a gas bubble through a liquid using the VoF method, with species transport across the bubble interface according to the CST Equation. The flow domain had a height of 10 cm and a width of 5 cm. A bubble of the gas phase was initialised with a diameter of 5 mm at 1 cm above the bottom surface. The following table lists the fluid properties used:

Fluid	ρ [kg/m^3]	μ [kg/(m s)]	ν [m^2/s]	D [cm^2/s]	$H = C_g/C_l$
Water-glycerol	1205	9.45×10^{-3}	6.2241×10^{-5}	2.13×10^{-6}	33
O_2	1.122	1.824×10^{-5}	1.6257×10^{-5}	0.2085	

The concentration field was initialised with $C = C_g = 8\,\mathrm{mol/m^3}$.

The discretisation schemes used for the divergence terms is shown below:

```
1 divSchemes
2 {
3     div(rhoPhi,U)   Gauss linearUpwind grad(U);
4     div(phi,alpha)   Gauss vanLeer;
5     div(phirb,alpha) Gauss linear;
6     div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
7     div(phi,C)       Gauss vanLeer;
8     default   Gauss linear;
9 }
```

Listing 4: Excerpt from fvSchemes

Without the use of the van Leer divergence scheme as seen on line 7, the concentration was found to become unbounded over time.

In the final setup, the case was simulated using a deltaT = 0.00001. The simulation was finely resolved. The simulation time step was increased further to deltaT = 0.0001, where again the simulation was resolved with a maximum CFL number of 0.083. But when the deltaT was changed to 0.001, the simulation crashed, where some floating point errors were reported and the maximum CFL number was approximately 0.83. The crash occurred during the solution of the CST Equation. Thus we may conclude that for higher CFL numbers, the solution might diverge and there is a certain restriction on deltaT.

Grid Convergence Study

In order to study grid convergence, three mesh sizes were used. The given case had a mesh with 500 cells in the x-direction and 1000 elements in the y-direction. To satisfy the given requirement of having at least 20 grid points per bubble diameter, the coarsest mesh size used had 200 cells in the x-direction, and 400 cells in the y-direction. The given mesh was chosen to be the finest mesh. In order to simplify calculations, the mesh refinement ratio and aspect ratio were kept constant across all three meshes.

The mesh refinement ratio was calculated as:

$$r = \sqrt{\frac{500}{200}} = 1.58$$

The resolutions of the meshes used were:

Mesh #	Resolution
Mesh 1	500 × 1000
Mesh 2	316 × 632
Mesh 3	200 × 400

The variable of interest selected was the mass transfer coefficient κ, the values of which were determined at various times for all three meshes and used to calculate the grid convergence index. The mass transfer coefficient was calculated using ParaView [1].

Time [s]	κ for Mesh1	κ for Mesh2	κ for Mesh3	$GCI^{32}/r^p \cdot GCI^{21}$
0.01	0.3206	0.2893	0.1929	1.108
0.02	0.01457	0.01599	0.03101	0.9127
0.04	0.01151	0.01081	0.01693	1.065
0.05	0.008333	0.007166	0.009297	1.163
0.1	0.003454	0.003414	0.003818	1.012
0.2	0.0009420	0.0008811	0.001143	1.069
0.25	0.0006430	0.0006020	0.0007359	1.0681

The grid convergence index was calculated as follows. First, the order of convergence was calculated using equation (1) [2].

$$p = \frac{1}{ln(r)} \times \left(ln \left| \frac{\Phi_3 - \Phi_2}{\Phi_2 - \Phi_1} \right| \right) \tag{1}$$

In equation (1), Φ is the variable of interest (κ in this case), and the subscripts 1, 2 and 3 refer respectively to meshes in decreasing order of refinement, i.e. 1 refers to the finest mesh, and 3 refers to the coarsest.

Next, the relative error and grid convergence index (GCI) were calculated using equations (2) to (5) [2],

$$\epsilon^{21} = \left| \frac{\Phi_1 - \Phi_2}{\Phi_1} \right| \tag{2}$$

$$\epsilon^{32} = \left| \frac{\Phi_2 - \Phi_3}{\Phi_2} \right| \tag{3}$$

$$GCI^{21} = \frac{F_s \cdot \epsilon^{21}}{r^p - 1} \tag{4}$$

$$GCI^{32} = \frac{F_s \cdot \epsilon^{32}}{r^p - 1} \tag{5}$$

where $F_s = 1.25$.

The decision of which mesh to use was made based on whether the meshes yield solutions that are in the asymptotic range of convergence. This was checked using (6).

$$\frac{GCI^{32}}{r^p \cdot GCI^{21}} \approx 1 \tag{6}$$

Equation (6) was found to hold for Meshes 1, 2 and 3, and hence Mesh 2 was chosen to carry out further calculations and post-processing.

Boundedness of Numerical Simulation

Numerical solutions are said to be bounded when the numerical values and physical properties calculated from the numerical solutions are within reasonable bounded range of the different phyiscal properties. In the case of fluid simulations, some common examples of reasonable boundaries of different physical properties are as follows.

- Density values that are larger than 0 (non-negative density)
- Concentration values between 0 and 1 or 0% to 100%
- VOF values between 0 and 1

Therefore, boundedness of the numerical solution is also an important metric of how physically realistic the numerical scheme is, especially in industrial and real-life applications where a numerical simulation that does not reflect the behaviour of physical systems could result in undesirable to catastrophic effects. An unbounded numerical solution generally stems from incorrect numerical schemes chosen or errors in the modelling process.

To check the boundedness of our numerical simulation, we look into our simulated α values which are summarized in the following plots.

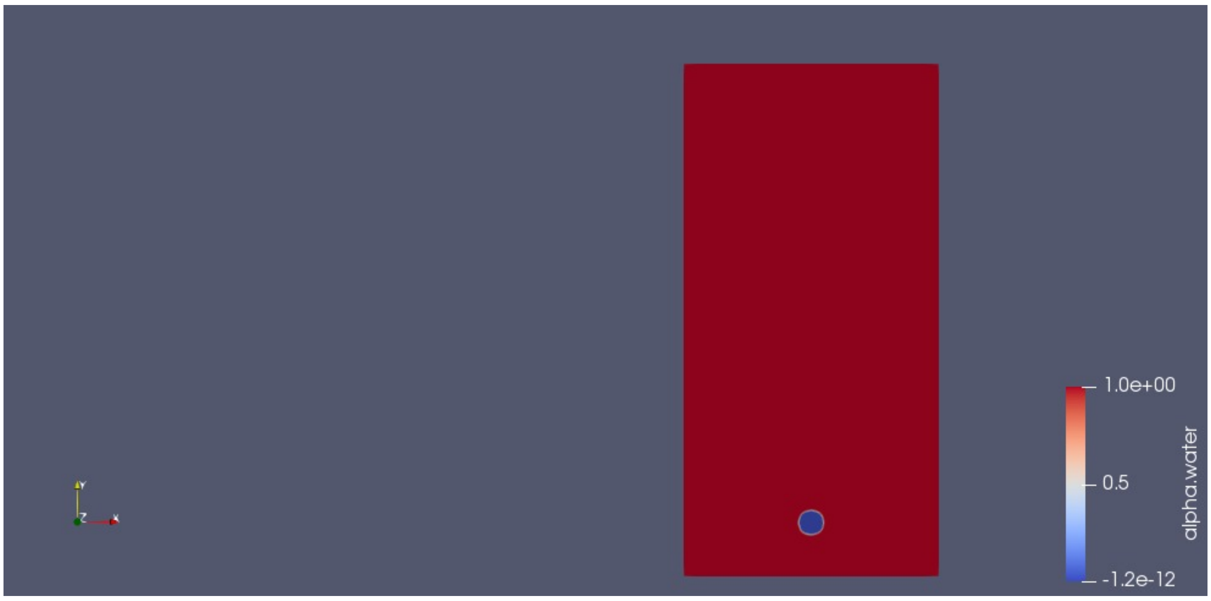

Figure 1: Contour plot of α values over the simulation domain at t = 0.1 s.

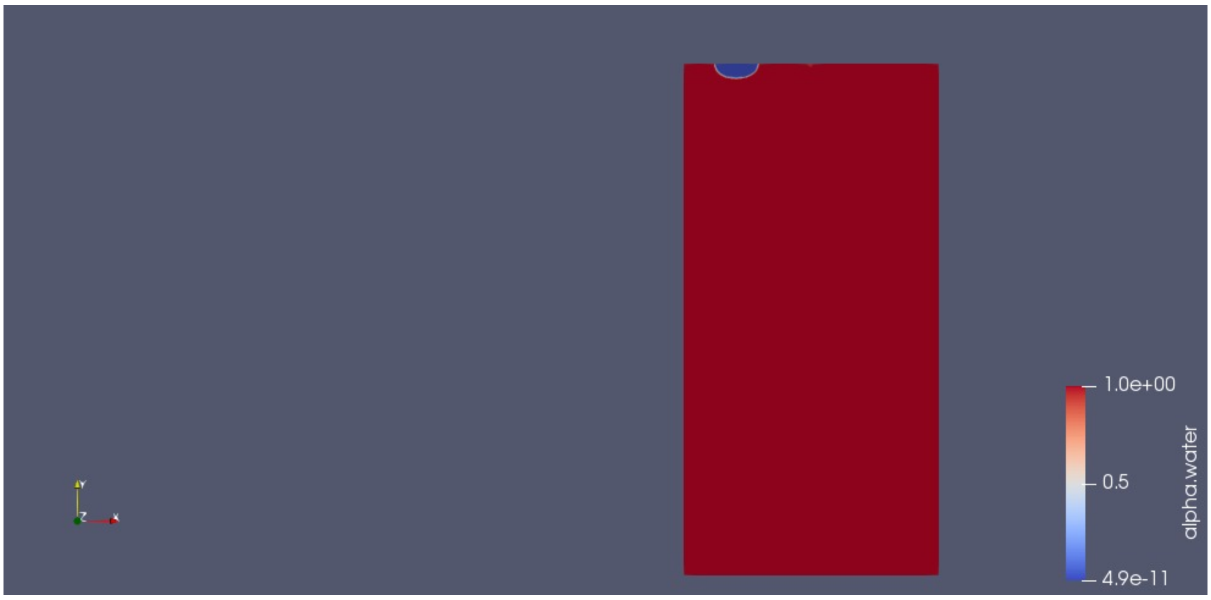

Figure 2: Contour plot of α values over the simulation domain at t = 1.2 s.

We see here that the α values throughout the domain are bounded between 0 for regions inside the bubble and 1 for regions outside of the bubble.

The simulated α values are always bounded between 0 and 1 throughout the whole simulation domain and throughout the different time values. Therefore, we can conclude that our numerical simulation is bounded.

Results and Interpretation

The concentration field at various times during the simulation are shown below. The colourmaps have been rescaled in each contour plot, in order to better illustrate the distribution of the species throughout the domain at each time step.

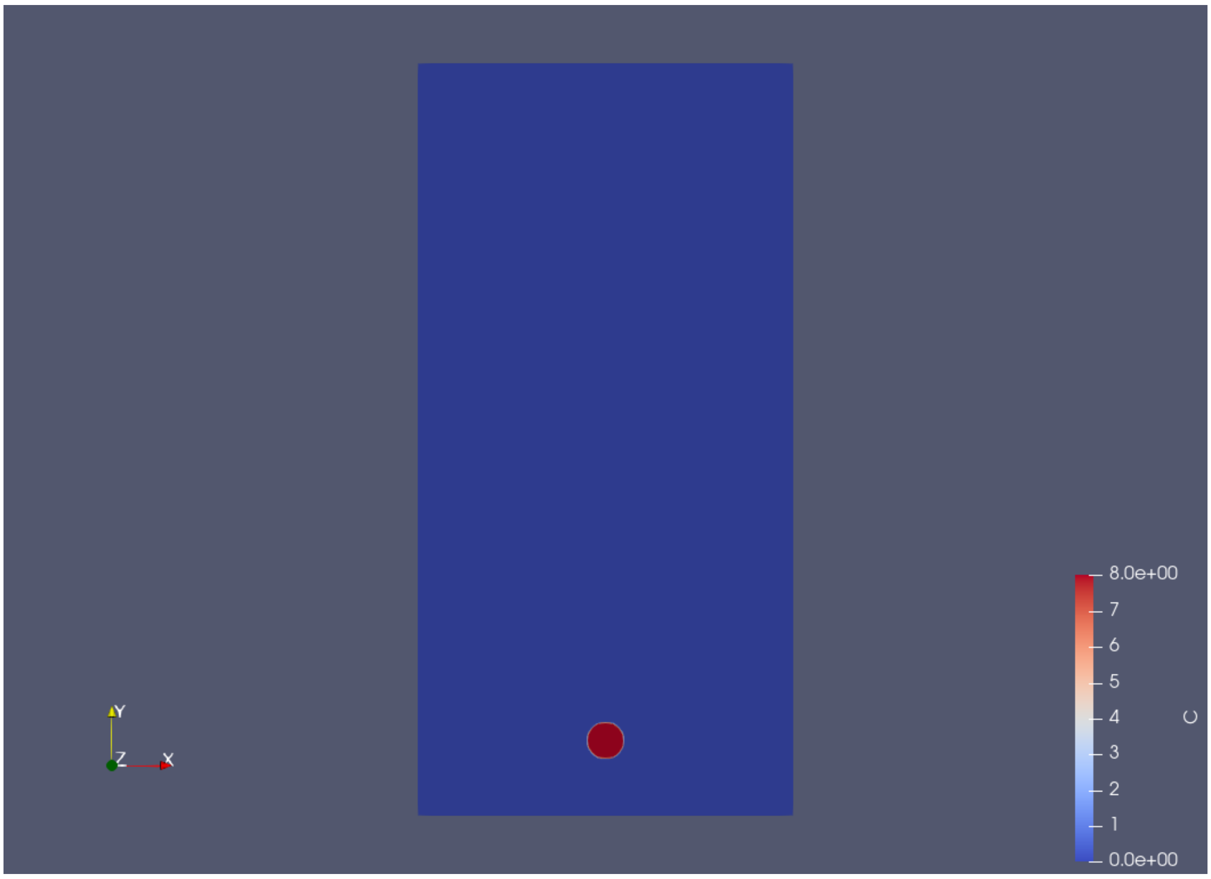

Figure 3: Concentration field at t = 0 s

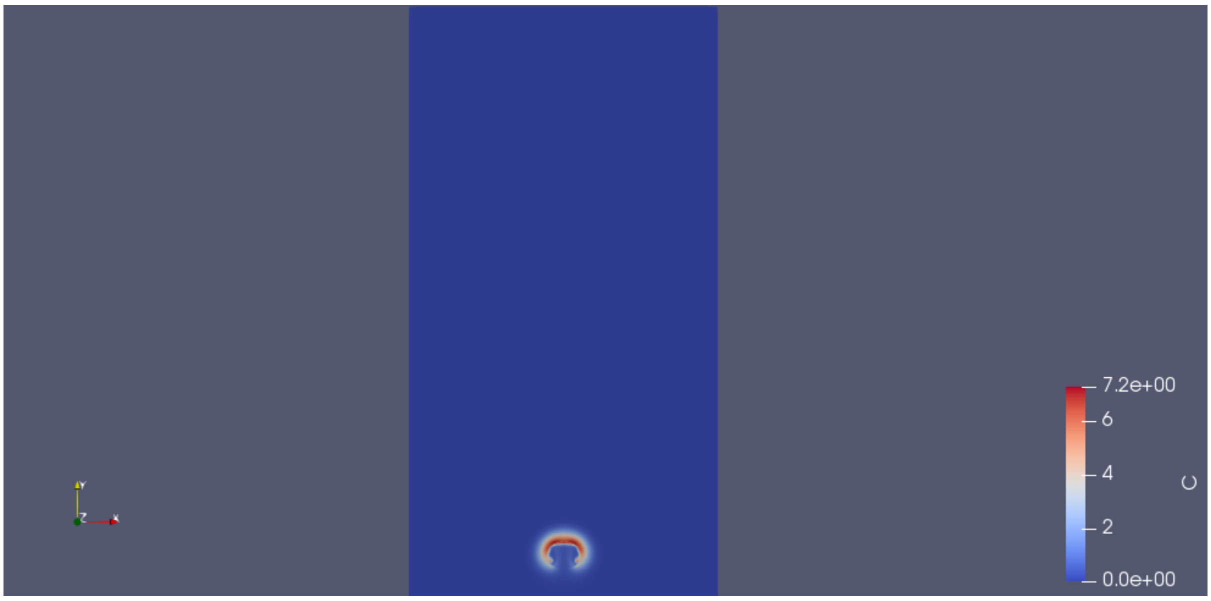

Figure 4: Concentration field at t = 0.02 s

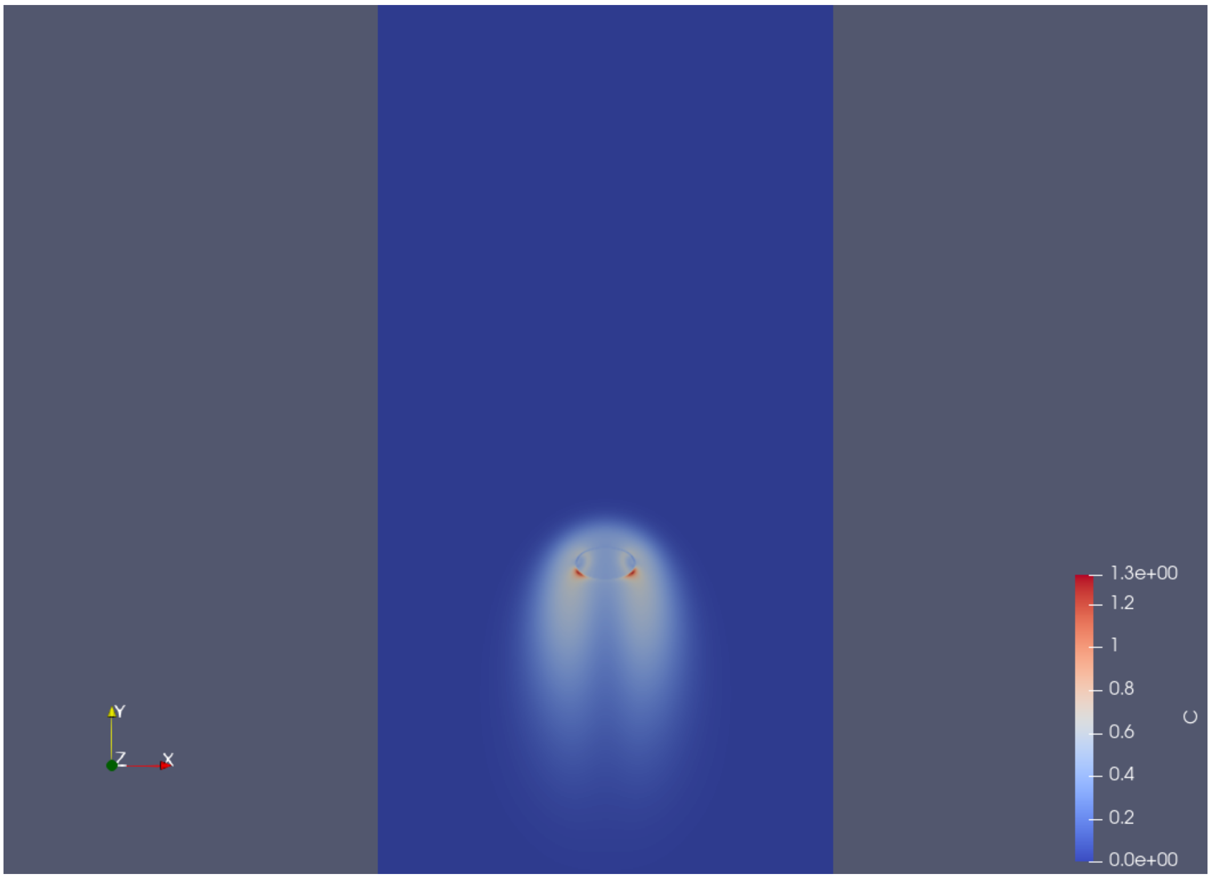

Figure 5: Concentration field at t = 0.2 s

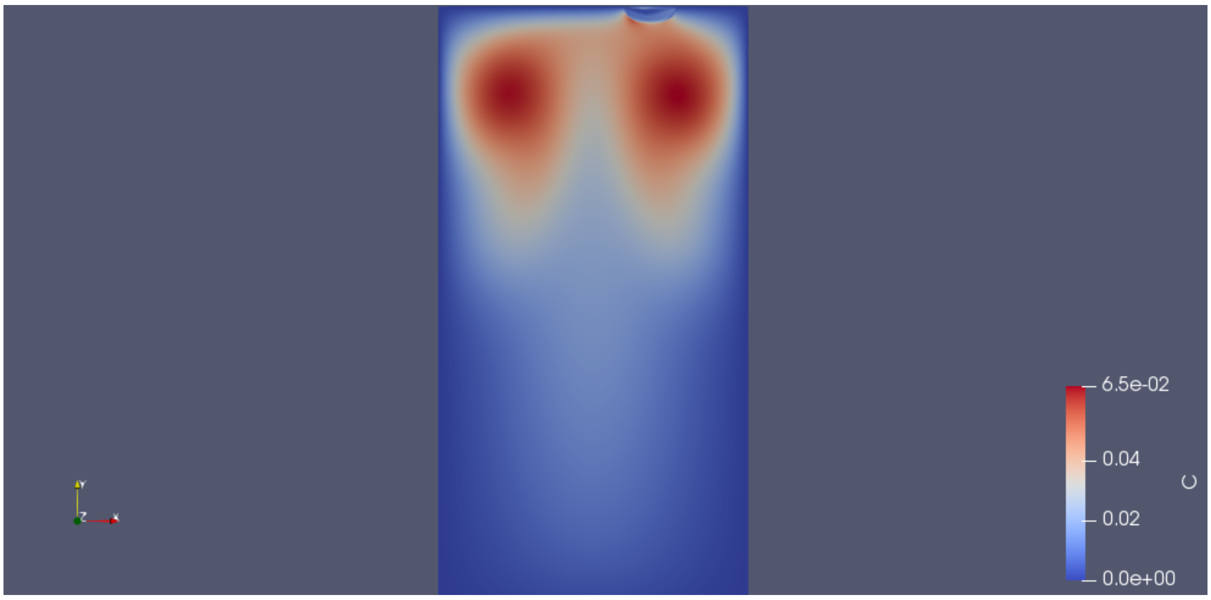

Figure 6: Concentration field at t = 1.5 s

As can be seen from the above contour plots, the concentration of species is initially present only in the bubble (gas phase), and over time the species is transported across the interface according to the CST equation. The bubble rises through the domain due to buoyancy, leading to the particular distribution of the species at the final time.

Bibliography

[1] Abhijit Rao. Ambit of multiphase cfd in modelling transport processes related to oil spill scenario and microfluidics. 2015.

[2] NPARC Alliance CFD Verification and Validation Web Site. Examining Spatial (Grid) Convergence.